Comments on "Distributed identification of the most critical node for average consensus"

Alexander Bertrand, *Senior Member, IEEE*

***Abstract*—In a previous paper, Liu et al. have presented a distributed method to rank the nodes of a network according to their importance for maintaining a fast average consensus within the network. Their method essentially estimates the decrease in algebraic connectivity for each possible node removal, based on a distributed estimation of the Fiedler vector. In this comment correspondence, we argue that their approach is limited to certain parameter ranges in the average consensus algorithm, and we briefly comment on how the framework can be extended accordingly. We also point out that their proposed algorithm for distributed Fiedler vector computation is essentially a special case of an earlier proposed algorithm, and in fact a numerically unstable version thereof. Finally, we correct some statements in their paper.**

***Index Terms*—Fiedler vector, distributed estimation, consensus**

I. Introduction

In [1], Liu et al. propose an algorithm to rank the nodes of a network based on the decrease in the convergence speed of the average consensus algorithm (ACA) for each possible node removal. They claim that this decrease is proportional to the change in the algebraic connectivity (AlCo) of the network, and they explain how the latter can be estimated using the Fiedler vector. In Section II, we explain that their method is only valid for specific parameter settings in the ACA, and may lead to incorrect node ranking for other parameter ranges. We argue that the largest eigenvalue of the graph Laplacian often also plays an important role, and we briefly explain how this can be included in the framework of [1]. In Section III, we point out that the algorithm proposed by Liu et al. for the distributed computation of the Fiedler vector, is in fact a special case of the algorithm in [2]. Moreover, we show that their algorithm is a numerically unstable version of the latter, which makes it eventually diverge away from the Fiedler vector in finite-precision arithmetic. Finally, in Section IV, we correct three statements in [1].

Notation: We mainly adopt the same notation as in [1]. Equation numbers followed by an asterisk '*', e.g., (4*), refer to equations in [1], rather than equations in this comment correspondence.

II. Comments on node importance evaluation

A. Convergence speed of the ACA

In [1], Liu et al. assume an ACA with a Laplacian-based combination rule, i.e., the $n\times n$ combination matrix is chosen as $\mathbf{I}-\rho\mathbf{L}$, where $\mathbf{L}$ is the so-called graph Laplacian, and where ρ is a (well-chosen) positive number. Their aim is to identify the most critical node whose removal would cause the largest decrease in convergence speed of the ACA. For each possible node removal, they estimate the change in the AlCo to quantify the change in convergence speed of the ACA. However, in [3], it has been shown that the asymptotic convergence factor for Laplacian-based ACA is equal to

$$\gamma = \max\left\{1-\rho\lambda_2, \rho\lambda_n - 1\right\} \tag{1}$$

where λ_n and λ_2 denote the largest and one-but-smallest eigenvalue of $\mathbf{L}$ (λ_2 is referred to as the AlCo). This demonstrates that, depending on the value of ρ, the convergence speed of the ACA is not always directly proportional to the AlCo, and hence the framework in [1] does not always apply. This observation was not made in [1], and unawareness of this fact may lead to wrong conclusions or node rankings, in particular when ρ is optimized for fast convergence. Indeed, in Section VI.A* in [1], the authors themselves set ρ to

$$\hat{\rho} = \frac{2}{\lambda_n + \lambda_2}\ , \tag{2}$$

to maximize (1). In this case, $\gamma = (\lambda_n - \lambda_2)/(\lambda_n + \lambda_2)$, and hence *both* λ_n and λ_2 can have an impact on the convergence speed of the ACA [4]. Furthermore, if $\rho > \hat{\rho}$, the AlCo has no impact whatsoever.

B. Proposed modification

The framework proposed in [1] can be extended to also include the effect of λ_n. Indeed, the derivation based on λ_2 and its corresponding eigenvector $\boldsymbol{\mu}$ (a.k.a. the Fiedler vector), can be easily modified for λ_n and its corresponding eigenvector $\mathbf{v}$. Furthermore, as shown in [4], both λ_n and $\mathbf{v}$ can also be computed in a distributed fashion. In fact, [4] aims to maximize γ by updating the edge weights in $\mathbf{L}$ based on a distributed computation of λ_n, λ_2, $\mathbf{v}$, and $\boldsymbol{\mu}$. The same ingredients are indeed needed to identify the critical nodes in [1].

Based on the derivation in [1] the counterpart of (4*) to quantify the change of λ_n (instead of λ_2) after removing node i is found as

$$\beta_i = \frac{\mathbf{v}_0^T\boldsymbol{\Delta}_i\mathbf{v}_i}{\mathbf{v}_0^T\mathbf{v}_i} \tag{3}$$

where the subscript i indicates a parameter after removal of node i, subscript 0 indicates a parameter before removing any node, and $\boldsymbol{\Delta}_i$ refers to the perturbation matrix on $\mathbf{L}_0$ corresponding to removal of node i, i.e., $\mathbf{L}_i = \mathbf{L}_0 - \boldsymbol{\Delta}_i$ (see [1]). We can then define three estimators of (3), similar to those proposed in [1], but now based on the eigenvector $\mathbf{v}$ instead of the Fiedler vector $\boldsymbol{\mu}$: the myopic estimator[1] (compare to (6*))

$$\hat{\beta}_i^{ME} = \frac{\mathbf{v}_0^T\boldsymbol{\Delta}_i\mathbf{v}_0}{\mathbf{v}_0^T\mathbf{v}_0}\ , \tag{4}$$

the zero-entry estimator (compare to (11*))

$$\hat{\beta}_i^{ZE} = \frac{\mathbf{v}_0^T\boldsymbol{\Delta}_i(\mathbf{v}_0 - v_{0,i}\,\mathbf{e}_i)}{\mathbf{v}_0^T(\mathbf{v}_0 - v_{0,i}\,\mathbf{e}_i)}\ , \tag{5}$$

and the spectrum-based estimator (compare to (17*))

$$\hat{\beta}_i^{SE} = \frac{\mathbf{v}_0^T\boldsymbol{\Delta}_i\mathbf{L}_i(\mathbf{v}_0 - v_{0,i}\,\mathbf{e}_i)}{\mathbf{v}_0^T(\mathbf{v}_0 - v_{0,i}\,\mathbf{e}_i)} \tag{6}$$

where $\mathbf{e}_i$ is the i-th column of the identity matrix. The matrix $\mathbf{C}_i$ in (17*) is replaced with the matrix $\mathbf{L}_i$ in (6), as $\mathbf{v}_i$ is the principal eigenvector of $\mathbf{L}_i$ (we refer to [1] for more details).

We now assume that λ_n and λ_2 are known (they are obtained as a by-product in the distributed computation of $\boldsymbol{\mu}$ and $\boldsymbol{v}$ [4]).

[1]The myopic estimator (6*) derived in [1] can be shown to coincide with the derivative of the AlCo with respect to a change of $\mathbf{L}_0$ in the direction $-\boldsymbol{\Delta}_i$ [4], [5]. It has also been referred to as Fiedler-vector centrality in [6].

The author acknowledges the financial support of the KU Leuven Research Council BOF/STG-14-005, C14/16/057, and CoE PFV/10/002 (OPTEC), Research Project FWO G.0931.14 & G0D7516N, and HANDiCAMS. The project HANDiCAMS acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number: 323944. The scientific responsibility is assumed by the author.

The author is with KU Leuven, Department of Electrical Engineering (ESAT), Stadius Center for Dynamical Systems, Signal Processing and Data Analytics, Kasteelpark Arenberg 10, box 2446, 3001 Leuven, Belgium (e-mail: alexander.bertrand@esat.kuleuven.be).

Based on (1), we propose the following modification to [1]:

1) If $\rho\lambda_n - 1 \ll 1 - \rho\lambda_2$, use the λ_2-estimators (6*), (11*), or (17*) as in [1].
2) If $\rho\lambda_n - 1 \gg 1 - \rho\lambda_2$, replace (6*), (11*), and (17*) in [1] with the λ_n-estimators (4), (5), and (6), respectively.
3) If $\rho \approx \hat{\rho}$, i.e., if a node removal could reverse the inequality in any of the above cases, then score that node according to the change in (1), using both the λ_2- *and* λ_n-estimator.

It is noted that the third case always yields the fastest convergence of the ACA, i.e., the first two cases are typically avoided in practice.

Remark: In [1], it was proposed to discard the denominators in the three estimators, as these are (almost) the same for all nodes, and will therefore not influence the node ranking. This still holds in case 1 or case 2, but not if $\rho \approx \hat{\rho}$ (case 3), since the absolute difference between the λ_2- and λ_n-estimators will determine the node ranking. In this case, a normalized estimate of $\mathbf{v}_0$ and $\boldsymbol{\mu}_0$ will have to be computed, e.g., using the distributed algorithm in [4].

III. Comments on distributed Fiedler vector estimation

In this section, we explain that the distributed algorithm proposed by Liu et al. in [1] to compute the Fiedler vector is actually a special case of the algorithm in [2], up to some heuristic differences, and except for a crucial numerical instability in the version of [1]. Liu et al. discuss [2], but apparently did not observe this equivalence.

To estimate the Fiedler vector, Liu et al. propose to use a power iteration (PI) $\mathbf{z}(k+1) = \mathbf{C}\mathbf{z}(k)$ with

$$\mathbf{C} = \mathbf{I} - \delta\mathbf{L} - \frac{1}{n}\mathbf{1}\mathbf{1}^T \tag{7}$$

where $\mathbf{1}$ denotes the all-ones vector and δ is a small positive number. Since $\mathbf{L}$ always has a zero eigenvalue ($\lambda_1 = 0$) with eigenvector equal to $\mathbf{1}$, the last term in (7) deflates the matrix $\mathbf{C}$ such that the maximal eigenvalue of $\mathbf{C}$ becomes $1-\delta\lambda_2$ (for sufficiently small δ), and hence the PI would converge to the Fiedler vector, instead of the $\mathbf{1}$ vector. Since the last term in (7) hampers a distributed implementation of the PI, the authors propose to initialize the PI with a vector that is orthogonal to $\mathbf{1}$, i.e., a zero-mean vector. Since $\mathbf{1}$ is in the nullspace of $\mathbf{L}$, such a vector can be obtained by multiplying any random vector with $\mathbf{L}$. One can then apply the PI with the matrix $\overline{\mathbf{C}} = \mathbf{I}-\delta\mathbf{L}$, since $\mathbf{1}^T\mathbf{z}(k)$ will remain zero over all iterations k.

The distributed algorithm in [2] to compute the Fiedler vector also performs a PI with $\overline{\mathbf{C}} = \mathbf{I} - \delta\mathbf{L}$, but in addition it applies a single multiplication with $\mathbf{L}$ after every N PI steps, where N is set by the user. This interleaved multiplication with $\mathbf{L}$ is referred to as the 'mean subtraction step' in [2], and its purpose is indeed also to orthogonalize the most recent estimate with respect to $\mathbf{1}$, to guide the estimate away from the principal eigenvector of $\overline{\mathbf{C}}$, towards the Fiedler vector.

The algorithm proposed in [1] is therefore a special case, where $N = \infty$, since the mean subtraction step is only performed once, i.e., during the initialization of the algorithm. However, in [2], it is stated that N should not be chosen too large, as the estimate will slowly but surely loose its initial zero-mean property due to inevitable rounding errors. Due to this critical numerical issue, the estimate in the version of Liu et al. will eventually diverge away from the Fiedler vector, and converge towards the dominant eigenvector of $\overline{\mathbf{C}}$, i.e., $\mathbf{1}$. This instability will always occur in practice after some time (depending on the size of these rounding errors), even in Matlab machine precision. This is demonstrated with a Monte-Carlo simulation shown in Fig. 1 for random toy networks of 10 to 20 nodes (the vector is normalized in every iteration for the sake of visualization).

In addition to the particular choice of N, the algorithm in [1] also slightly differs in the heuristic protocol to avoid the vector norm to shrink to 0 during the PI. The protocol used in [1] makes the estimate only converge up to a time-varying scaling (which may be sufficient in some applications). The protocol used in [2] lets the algorithm converge to a stable Fiedler vector estimate with a static norm, allowing to, e.g., apply normalization algorithms [4]. However, this stabilization requires a parallel diffusion algorithm, which adds some additional complexity and communication overhead.

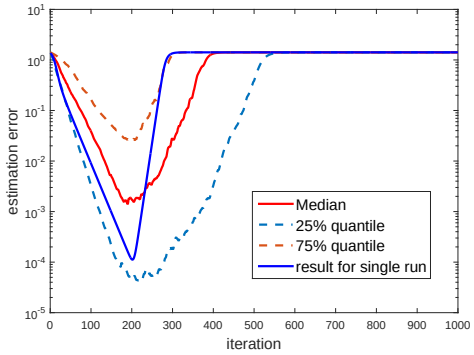

Fig. 1. Fiedler vector mean-squared estimation errors over 300 Monte Carlo runs of the algorithm in [1] + result of a single run.

IV. Additional corrections

Correction 1: Liu et al. claim that the algorithm in [2] converges exponentially at a rate $1 - \delta\lambda_2$, whereas their own algorithm has a rate of $\frac{1-\delta\lambda_2}{1-\delta\lambda_3}$, which for certain values of δ could then outperform the former. This claim is factually incorrect: in Remark III in [2], it is explained that the rate of the algorithm in [2] is not $1 - \delta\lambda_2$, but

$$\left(\frac{\lambda_2}{\lambda_3}\right)^{\frac{1}{N}} \left(\frac{1-\delta\lambda_2}{1-\delta\lambda_3}\right)^{\frac{N}{N-1}} \tag{8}$$

and that this is equal to $\frac{1-\delta\lambda_2}{1-\delta\lambda_3}$ when $N \to \infty$. This is indeed the same rate as in [1], as the latter is a special case of [2] with $N \to \infty$.

Correction 2: In Section V.C* in [1], Liu et al. state that their method "involves algebraic connectivity estimation and Fiedler vector computation, while [2] only focus on the latter". This is an incorrect statement: in [2], each node also obtains an estimate of the algebraic connectivity, as mentioned in Remark V in [2].

Correction 3: In [1], Liu et al. state that the algorithm in [2] is "centralized because of the need to designate a global beacon node". This is incorrect, as this beacon node does not affect the distributed nature of the algorithm. Its role is to disseminate a network-wide rescaling factor, using a simple flooding protocol. Any arbitrary node can act as a beacon node, and this can be negotiated using simple distributed protocols, e.g., based on a random number generator and a minimum-consensus algorithm. This selection only happens once during initialization, and a (neighboring) node can immediately take over its role in case of a node failure. In fact, the triggering event in [1] can also be viewed as a particular way to select a beacon node.

References

[1] H. Liu, X. Cao, J. He, P. Cheng, C. Li, J. Chen, and Y. Sun, "Distributed identification of the most critical node for average consensus," *IEEE Trans. Signal Processing*, vol. 63, no. 16, pp. 4315–4328, Aug 2015.
[2] A. Bertrand and M. Moonen, "Distributed computation of the Fiedler vector with application to topology inference in ad hoc networks," *Signal Processing*, vol. 93, no. 5, pp. 1106–1117, May 2013.
[3] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, no. 1, pp. 65 – 78, 2004.
[4] A. Bertrand and M. Moonen, "Topology-aware distributed adaptation of Laplacian weights for in-network averaging," in *Proc. European signal processing conference (EUSIPCO)*, Marrakech, Morocco, Sep. 2013.
[5] A. Ghosh and S. Boyd, "Growing well-connected graphs," in *Proc. IEEE Conference on Decision and Control*, Dec. 2006, pp. 6605–6611.
[6] P.-Y. Chen and A. Hero, "Local Fiedler vector centrality for detection of deep and overlapping communities in networks," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, May 2014, pp. 1120–1124.